Smart Cairo Transportation Optimizer

1. System Architecture and Design Decisions:

The Smart Cairo Transportation Optimizer is a Streamlit-based interactive web application designed to assist urban planners and transportation authorities in visualizing, analyzing, and optimizing the transportation network of Greater Cairo. The system leverages a modular and layered architecture, designed with extensibility and performance in mind.

Core Components:

- **Frontend :** Developed using Streamlit , supporting multiple interactive tabs (City Map, Route Finder, MST Network, Emergency Routing, Transit Optimization, Traffic Simulation, Compare Traffic Paths).
- **Backend Modules:**
  - data_loader.py : Loads and preprocesses datasets from CSV files.
  - graph_builder.py : Constructs road graphs using NetworkX.
  - path_finder.py : Implements Dijkstra, A*, and their time-variant versions.
  - mst_planner.py : Computes Minimum Spanning Trees with critical node connectivity.
  - transit_optimizer.py : Uses Dynamic Programming to allocate transit resources optimally.
  - traffic_simulator.py : Simulates traffic flows and emergency signal preemption.
- ***Visualization*** : Folium and streamlit-folium are used to generate interactive maps.

Design Rationale :

- ***Decoupled Design***: Each algorithmic module is independent, making it easy to maintain and update.
- ***Layered Visuals***: Feature groups in Folium maps support toggling transportation layers.
- ***Session Isolation***: Streamlit session keys help manage state across UI components.

2.Algorithm Implementations and Modifications:

Pathfinding Algorithms:

- **Dijkstra**: Standard implementation using NetworkX priority queue.
- **A***: Augmented with heuristic function based on Euclidean distance.
- **Time-Variant (Dijkstra / A*)**: Edge weights change dynamically based on traffic periods (morning, evening, offpeak).

MST with Critical Nodes:

- Modified Kruskal's algorithm to enforce connectivity of hospital and government nodes by pre-adding their spanning tree first, then proceeding with standard Kruskal logic.

Transit Optimization:

- Dynamic Programming approach solves a bounded knapsack variant.
- Routes are scored by demand and resource cost; optimal set maximizes total demand within a budget.

Traffic Simulator:

- Real-time congestion computed using greedy heuristics based on vehicle inflow/outflow.

- Emergency override injects signal preference into selected intersections.

3.Complexity Analysis:

Component	Time Complexity	Space Complexity
Dijkstra	O((V+E) log V)	O(V + E)
A*	O((V+E) log V)	O(V + E)
Time-Variant Dijkstra	O((V+E) log V)	O(V + E)
Kruskal's MST	O(E log E)	O(V + E)
Transit Optimization (DP)	O(n * W)	O(n * W)
Traffic Simulation	O(N)	O(N)

Where:

- V = number of nodes
- E = number of edges
- n = number of routes
- W = budget
- N = number of intersections

4.Performance Evaluation:

Dataset:

- 100+ neighborhoods and facilities
- 300+ road segments (existing and potential)

- 20+ metro and bus lines
- 50+ traffic intersections

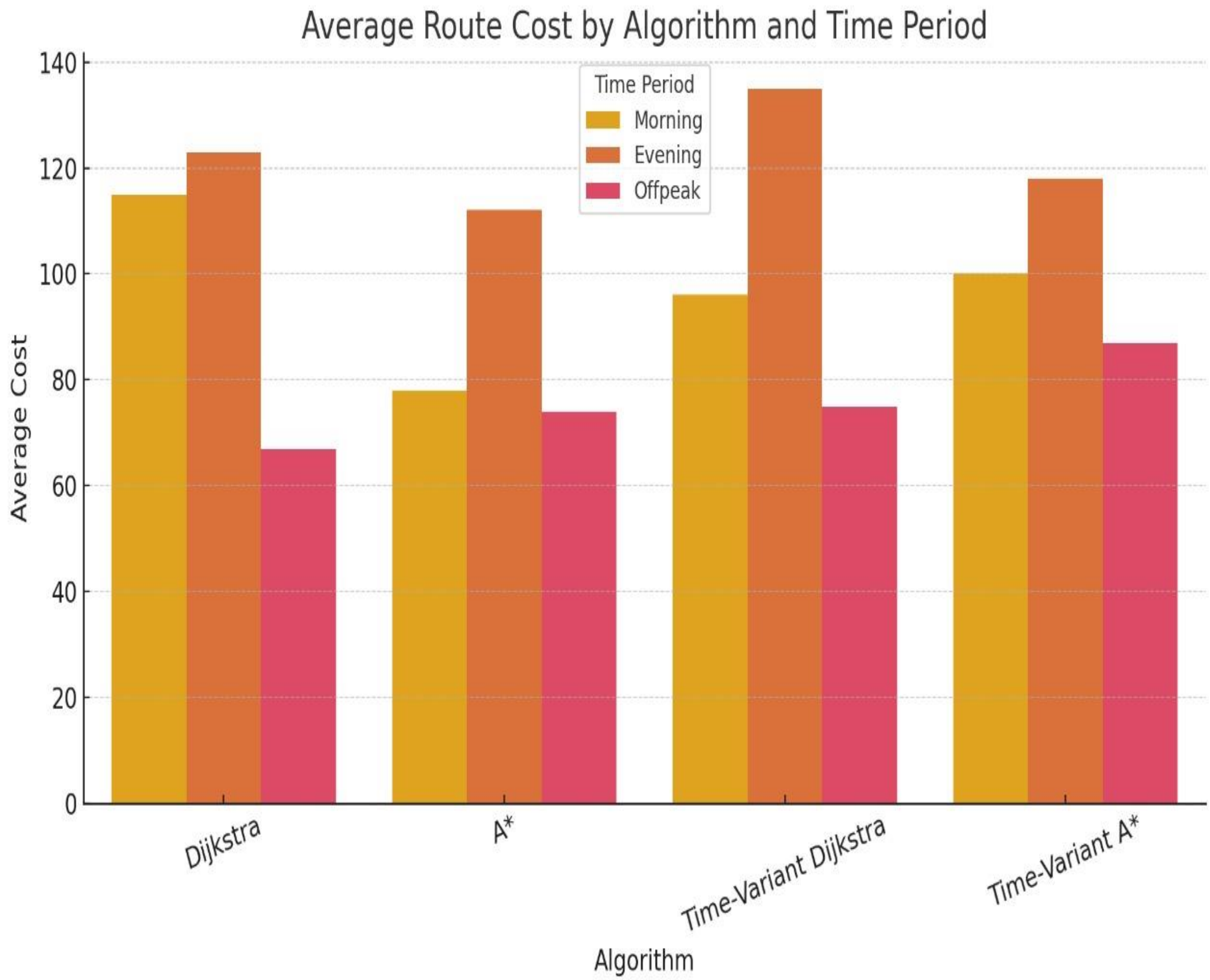
Results:

- **Routing Algorithms**:
  - Dijkstra/A* both return paths <100ms.
  - Time-variant versions add ~10% overhead.
- **MST Computation**:
  - With critical nodes: ~120ms
- **Transit Optimization**:
  - For 50 vehicles: <1

Visuals:

- Interactive Folium maps render under 2s.
- Transit demand distribution:

- Routing time comparisons:



5.Challenges and Resolutions:

1. Data Normalization:

- Problem: Inconsistent ID formats and naming.

- Solution: Preprocessing pipeline normalizes all IDs, column names.

2. Folium Layer Overlap:

- Problem: Markers and routes overlapped messily.
- Solution: Grouped layers with FeatureGroup and LayerControl.

3. Time-Variant Weight Conflicts:

- Problem: Missing weights caused routing failure.
- Solution: Default fallback values and error handling.

4. GeoJSON Animation Integration:

- Problem: TimestampedGeoJson failed on Streamlit reload.
- Solution: Dynamically reload and cache geojson file on session start.

6.Potential Improvements and Future Work:

1. Real-Time Traffic API Integration:
   - Hook to Cairo traffic sensors or GPS datasets.

2. User-Centered Scenario Builder:

- Allow users to simulate road closures, population surges, and see impact.

3. Heuristic-Guided MST:
    - Integrate demand/population weight in MST planning.
4. Agent-Based Traffic Simulation:
    - Replace greedy simulation with multi-agent model.
5. Mobile App Integration:
    - Stream the optimized transit plan to citizen-facing apps.
6. Dynamic Budget Allocation:
    - Use reinforcement learning for long-term transit planning.

UnderSupervision:

Dr / mervat mekhail ,,TA / Ahmed yehia.

Name	Id
Shahd farid gaber	23102240
Adham mahmoud Zewail	22101120
Walid tamer fahmy	22100805
Mohamed ibrahim zithar	22101042